IN THE UNITED STATES PATENT AND TRADEMARK OFFICE

In re the Application of: Sowmya Subramanian, et al. **Serial No.**: 09/872,589 **Filed**: May 31, 2001 **For**: METHODS, SYSTEMS, AND ARTICLES OF MANUFACTURE FOR PREFABRICATING AN INFORMATION PAGE	**Group Art Unit**: 2178 **Examiner**: Stork, Kyle R. **Confirmation No.**: 2629

DECLARATION OF SOWMYA SUBRAMANIAN ET AL. PURSUANT TO 37 C.F.R. § 1.131

Commissioner for Patents
P.O. Box 1450
Alexandria, VA 22313-1450

Dear Examiner Stork:

We, Sowmya Subramanian, Ramu Sunkara, Kunal Kapur, Anthony Lai, Sarim Siddiqui, Sunny Wong, and Hyun-Sik Byun, declare as follows:

1. We are the named inventors of the above-identified application.

2. Prior to December 18, 2000, we conceived and actually reduced to practice, in the United States, the invention disclosed and claimed in the above-identified application.

3. We have reviewed the currently pending claims of the above-identified application (i.e., claims 1-91), and to the best of our recollection, ***we believe that we had a definite and complete idea***

of, and had actually reduced to practice, the designs embodying all of the elements of claims 1-91 prior to December 18, 2000. Independent claims 1, 23, 49, 58, 63, 70, and 71 are set forth below:

1. A method for prefabricating an information page, comprising:

prefabricating a first page in accordance with a definable prefabrication policy to produce a first prefabricated page, wherein the prefabricating is not in response to a request for the first page by a user;

receiving an information request;

determining if the information request corresponds to the first page;

providing the first prefabricated page if the information request corresponds to the first page; and

dynamically fabricating a second page if the information request corresponds to the second page;

wherein the act of prefabricating the first page comprises querying a database to obtain cached data, processing the data received from the database, and packaging information associated with the data in a prescribed format.

23. A system for prefabricating information, comprising:

a prefabricator configured to prefabricate a first page to produce a first prefabricated page, wherein the first page is not prefabricated by the prefabricator in response to a request for the first page by a user, and wherein the act of prefabricating the first page comprises querying a database to obtain cached data, processing the data received from the database, and packaging information associated with the data in a prescribed format;

an interceptor to intercept an information request, the interceptor logically interposed between a user interface and a computer

application, the interceptor providing the first prefabricated page if the information request corresponds to the first page and dynamically fabricating a second page if the information request corresponds to the second page.

49. A method for prefabricating information pages, comprising:

prefabricating a first page on a first node to produce a first prefabricated page, wherein the prefabricating is not in response to request for the first page by a user;

storing the first prefabricated page;

prefabricating a second page on a second node to produce a second prefabricated page;

storing the second prefabricated page;

receiving an information request;

providing the first prefabricated page if the information request corresponds to the first page; and

providing the second prefabricated page if the information request corresponds to the second page;

wherein the act of prefabricating the first page comprises querying a database to obtain cached data, processing the data received from the database, and packaging information associated with the data in a prescribed format.

58. (Previously Presented) A method for prefabricating an information page, comprising:

prefabricating a first page to produce a first prefabricated page , wherein the prefabricating is not in response to a request for the first page by a user;

receiving an information request from a user having a session identifier;

determining if the information request corresponds to the first page;

providing the first prefabricated page with the session identifier if the information request corresponds to the first page; and

dynamically fabricating a second page if the information request corresponds to the second page;

wherein the act of prefabricating the first page comprises querying a database to obtain cached data, processing the data received from the database, and packaging information associated with the data in a prescribed format.

63. (Previously Presented) A method for prefabricating an information page comprising:

obtaining one or more parameters that define how a page should be prefabricated; and

prefabricating a page based on the one or more parameters, wherein the prefabricating is not in response to a request for the page by a user, and wherein the act of prefabricating the page comprises querying a database to obtain cached data, processing the data received from the database, and packaging information associated with the data in a prescribed format.

70. (Previously Presented) A computer program product that includes a medium usable by a processor, the medium having stored thereon a sequence of instructions which, when executed by said processor, causes said processor to execute a process for prefabricating an information page, the process comprising:

prefabricating a first page in accordance with a definable prefabrication policy to produce a first prefabricated page, wherein the prefabricating is not in response to a request for the first page by a user;

receiving an information request;

determining if the information request corresponds to the first page;

providing the first prefabricated page if the information request corresponds to the first page; and

dynamically fabricating a second page if the information request corresponds to the second page;

wherein the act of prefabricating the first page comprises querying a database to obtain cached data, processing the data received from the database, and packaging information associated with the data in a prescribed format.

71. (Previously Presented) A computer program product that includes a medium usable by a processor, the medium having stored thereon a sequence of instructions which, when executed by said processor, causes said processor to execute a process for prefabricating an information page, the process comprising:

prefabricating a first page on a first node to produce a first prefabricated page , wherein the prefabricating is not in response to a request for the first page by a user;

storing the first prefabricated page;

prefabricating a second page on a second node to produce a second prefabricated page;

storing the second prefabricated page;

receiving an information request;

providing the first prefabricated page if the information request corresponds to the first page; and

providing the second prefabricated page if the information request corresponds to the second page;

wherein the act of prefabricating the first page comprises querying a database to obtain cached data, processing the data received from the database, and packaging information associated with the data in a prescribed format.

4. The set of documents attached as **Exhibit A** titled "***Oracle CRM 11i 'Hornet' Release: JTF Page Prefabrication***," and **Exhibit B** titled "***Design Specification for JTF Page Fabrication, CRM Application, 11i,***"evidences that we conceived of each and every element of each claim of the above-identified application and reduced it to practice before December 8, 1998. The document of **Exhibit A**, which was made prior to December 18, 2000, composed part of a design specification used internally by the assignee of the above-identified application. The document of **Exhibit B** is part of a presentation made internally within the company of the assignee describing analysis results based upon running an implementation of the invention. The specification and presentation material were not used earlier than one year prior to the filing date of the above-identified application.

5. With respect to claims 1 and 70:

Exhibit A shows a design specification for prefabricating an information page, that includes

(1) *prefabricating a first page in accordance with a definable prefabrication policy to produce a first prefabricated page, wherein the prefabricating is not in response to a request for the first page by a user* (particularly **p. 7, Section 3.1.1 - 3.1.2**, and **p. 13, Section 3.4**),

(2) *receiving an information request (particularly* ***p. 20, Section 3.9****),*

determining if the information request corresponds to the first page (particularly **p. 20, Section 3.9**),

(3) *providing the first prefabricated page if the information request corresponds to the first page* (particularly **p. 20, Section 3.9**), and

(4) *dynamically fabricating a second page if the information request corresponds to the second page* (particularly **p. 11, Section 3.2**, and **p. 19, Section 3.5**),

(5) *wherein the act of prefabricating the first page comprises querying a database to obtain cached data, processing the data received from the database, and packaging information associated with the data in a prescribed format* (particularly **p. 19, Section 3.6**, and **p. 20, Sections 3.8-3.9**).

In addition, **Exhibit B** is part of a presentation made internally within the company of the assignee describing analysis results based upon running an implementation of the invention that includes:

(1) *wherein the act of prefabricating the first page comprises querying a database to obtain cached data, processing the data received from the database, and packaging information associated with the data in a prescribed format*

(a) **p. 4, Section 2** "[t]he key idea behind page prefabrication is to create and store the HTTP Response for a URL request ***a priori***, *so that* ***when an user actually requests the URL it can be serviced immediately***"

(b) **p. 5 Exhibit B** shows the three major components of the design to be the prefabricator, the interceptor, and the cache where at least one of the embodiments of the claimed invention as well as **Exhibit B** are focusing upon the interceptor and the prefabricator. **Exhibit B** thus demonstrates that both the interceptor and the prefabricator may serve their respective functions without receiving any user requests.

(c) **Fig. 1** on **p. 4** and **Section 3.9** of **Exhibit B** on **pp. 20-21** show that the interceptor intercepts the URL requests either from the prefabricator or the user and thereby determines whether the request can be serviced from the cache or not. **Section 3.9** further

discloses the basic algorithm implemented by the interceptor to include both "[i]f from user" and "[i]f from spider" and "[i]n either case, response should also be T-d to the URL Collector . . . ." As such, the interceptor can clearly serve its intended function without referencing to any user requests, and **Exhibit B** thus discloses the aforementioned limitations.

(d) **Section 3** of **Exhibit B** shows that the prefabricator includes, to the extent pertinent to page prefabrication, a start loader, a benefit analyzer, a URL collector, and a page generator. **Fig. 3** on **p. 3** and **Section 3.3** on **p.12** of **Exhibit B** shows that the Benefit Analyzer receives PRBs to be processed from both the URL Collector and the Start Loader and determines the next set of pages to prefabricate. **Section 3.3** on **p.12** further discloses the selection algorithm of how the Benefit Analyzer determines which set of pages to prefabricate and what information must be made available to the Benefit Analyzer in order for the Benefit Analyzer to serve its intended functions. As **Section 3.3** clearly demonstrates, the selection algorithm of the Benefit Analyzer is based primarily upon the current fabrication rate of the system but need not rely on any user requests to make such determination.

(e) **Section 3.2** describes the Start Loader. Moreover, the logic for the Start Loader to implement the Runnable interface further describes how the Start Loader serves its intended functions. More specifically, the Start Loader uses the PRBEvaluateInterface implementer to generate all the PRBs that make up the initial workload; it then invokes the Benefit Analyzer to populate the PRBs; and the thread periodically wakes up to check if new PRBs need to be created and creates them. From this description, it is clear that the Start Loader requires no user requests to serve its intended purposes.

(f) **Section 3.4** describes the algorithm for the Page Request Feeder which performs page generation from the PRBs that have been processed by the Benefit Analyzer. As it can be seen from the description, the algorithm requires no user request to serve its intended function.

(g) **Sections 3.5** and **3.6** together with **Figs. 1-3** describe the functions of the URL Collector and the Page Generator. These paragraphs are absolutely silent on taking on any user requests in order to serve the functions. More specifically, **Section 3.5** describes that the

URL Collector functions like a web-crawler. It is well know in the art that a web-crawler typically functions in an automated manner. The Page Generator may also function free of any user requests because the Page Generator is interposed between the Page Request Feeder and the URL Collector, both of which may function independently from any user requests.

(2) *wherein the act of prefabricating the first page comprises querying a database to obtain cached data, processing the data received from the database, and packaging information associated with the data in a prescribed format*

(a) Particularly **Section 3.2** explicitly discloses that the functions of the default implementer of the PRBEvaluateInterface including "query the find_user, find_user_resp_groups and find_responsibility tables . . . ." Thus, **Exhibit B** clearly discloses the claimed limitation of "querying a database . . . ."

6. With respect to claim 23:

Exhibit A shows a design specification for a system for prefabricating information that includes

(1) *a prefabricator configured to prefabricate a first page to produce a first prefabricated page, wherein the first page is not prefabricated by the prefabricator in response to a request for the first page by a user, and wherein the act of prefabricating the first page comprises querying a database to obtain cached data, processing the data received from the database, and packaging information associated with the data in, a prescribed format* (particularly **p. 6, Section 3**),

(2) *an interceptor to intercept an information request, the interceptor logically interposed between a user interface and a computer application, the interceptor providing the first prefabricated page if the information request corresponds to the first page and dynamically fabricating a second page if the information request corresponds to the second page* (particularly **p. 20, Section 3.9**, and **p. 19, Section 3.5**).

In addition, **Exhibit B** is part of a presentation made internally within the company of the assignee describing analysis results based upon running an implementation of the invention that includes:

(1) *a prefabricator configured to prefabricate a first page to produce a first prefabricated page, wherein the first page is not prefabricated by the prefabricator in response to a request for the first page by a user, and wherein the act of prefabricating the first page comprises querying a database to obtain cached data, processing the data received from the database, and packaging information associated with the data in, a prescribed format*

(a) **p. 4, Section 2** "[t]he key idea behind page prefabrication is to create and store the HTTP Response for a URL request ***a priori***, *so that* ***when an user actually requests the URL it can be serviced immediately***"

(b) **p. 5 Exhibit B** shows the three major components of the design to be the prefabricator, the interceptor, and the cache where at least one of the embodiments of the claimed invention as well as **Exhibit B** are focusing upon the interceptor and the prefabricator. **Exhibit B** thus demonstrates that both the interceptor and the prefabricator may serve their respective functions without receiving any user requests.

(c) **Section 3** of **Exhibit B** shows that the prefabricator includes, to the extent pertinent to page prefabrication, a start loader, a benefit analyzer, a URL collector, and a page generator. **Fig. 3** on **p. 3** and **Section 3.3** on **p.12** of **Exhibit B** shows that the Benefit Analyzer receives PRBs to be processed from both the URL Collector and the Start Loader and determines the next set of pages to prefabricate. **Section 3.3** on **p.12** further discloses the selection algorithm of how the Benefit Analyzer determines which set of pages to prefabricate and what information must be made available to the Benefit Analyzer in order for the Benefit Analyzer to serve its intended functions. As **Section 3.3** clearly demonstrates, the selection algorithm of the Benefit Analyzer is based primarily upon the current fabrication rate of the system but need not rely on any user requests to make such determination.

(d) **Section 3.2** describes the Start Loader. Moreover, the logic for the Start Loader to implement the Runnable interface further describes how the Start Loader serves its intended

functions. More specifically, the Start Loader uses the PRBEvaluateInterface implementer to generate all the PRBs that make up the initial workload; it then invokes the Benefit Analyzer to populate the PRBs; and the thread periodically wakes up to check if new PRBs need to be created and creates them. From this description, it is clear that the Start Loader requires no user requests to serve its intended purposes.

(e) **Section 3.4** describes the algorithm for the Page Request Feeder which performs page generation from the PRBs that have been processed by the Benefit Analyzer. As it can be seen from the description, the algorithm requires no user request to serve its intended function.

(f) **Sections 3.5** and **3.6** together with **Figs. 1-3** describe the functions of the URL Collector and the Page Generator. These paragraphs are absolutely silent on taking on any user requests in order to serve the functions. More specifically, **Section 3.5** describes that the URL Collector functions like a web-crawler. It is well know in the art that a web-crawler typically functions in an automated manner. The Page Generator may also function free of any user requests because the Page Generator is interposed between the Page Request Feeder and the URL Collector, both of which may function independently from any user requests.

(2) *a prefabricator configured to prefabricate a first page to produce a first prefabricated page, wherein the first page is not prefabricated by the prefabricator in response to a request for the first page by a user, and wherein the act of prefabricating the first page comprises querying a database to obtain cached data, processing the data received from the database, and packaging information associated with the data in, a prescribed format*

(a) **Section 3.2** explicitly discloses that the functions of the default implementer of the PRBEvaluateInterface including "query the find_user, find_user_resp_groups and find_responsibility tables . . . ." Thus, we believe **Exhibit B** clearly discloses the claimed limitation of "querying a database . . . ."

7. With respect to claims 49 and 71:

Exhibit A shows a design specification for prefabricating information pages, which includes

(1) prefabricating a first page on a first node to produce a first prefabricated page, wherein the prefabricating is not in response to request for the first page by a user (particularly **p. 11, Section 3.2**, **p. 12, Section 3.3**, and **p. 13, Section 3.4**),

(2) storing the first prefabricated page (particularly **p. 13, Section 3.4**, **p. 19, Section 3.6**, and **p. 21, Section 3.10**),

(3) prefabricating a second page on a second node to produce a second prefabricated page (particularly **p. 12, Section 3.3**, and **p. 13, Section 3.4**),

(4) storing the second prefabricated page (particularly **p. 13, Section 3.4**, **p. 19, Section 3.6**, and **p. 21, Section 3.10**),

(5) receiving an information request (particularly **p. 20, Section 3.9**, and **p. 11, Section 3.2**),

(6) providing the first prefabricated page if the information request corresponds to the first page (particularly **p. 20, Section 3.9**), and

(7) providing the second prefabricated page if the information request corresponds to the second page (particularly **p. 20, Section 3.9**),

(8) wherein the act of prefabricating the first page comprises querying a database to obtain cached data, processing the data received from the database, and packaging information associated with the data in a prescribed format (particularly **p. 19, Section 3.6**, and **p. 20, Sections 3.8-3.9**).

In addition, **Exhibit B** is part of a presentation made internally within the company of the assignee describing analysis results based upon running an implementation of the invention that includes:

(1) *prefabricating a first page on a first node to produce a first prefabricated page, wherein the prefabricating is not in response to request for the first page by a user*

(a) **p. 4, Section 2** "[t]he key idea behind page prefabrication is to create and store the HTTP Response for a URL request ***a priori***, *so that* ***when an user actually requests the URL it can be serviced immediately***"

(b) **p. 5 Exhibit B** shows the three major components of the design to be the prefabricator, the interceptor, and the cache where at least one of the embodiments of the claimed invention as well as **Exhibit B** are focusing upon the interceptor and the prefabricator. **Exhibit B** thus demonstrates that both the interceptor and the prefabricator may serve their respective functions without receiving any user requests.

(c) **Fig. 1** on **p. 4** and **Section 3.9** of **Exhibit B** on **pp. 20-21** show that the interceptor intercepts the URL requests either from the prefabricator or the user and thereby determines whether the request can be serviced from the cache or not. **Section 3.9** further discloses the basic algorithm implemented by the interceptor to include both "[i]f from user" and "[i]f from spider" and "[i]n either case, response should also be T-d to the URL Collector . . . ." As such, the interceptor can clearly serve its intended function without referencing to any user requests, and **Exhibit B** thus discloses the aforementioned limitations.

(d) **Section 3** of **Exhibit B** shows that the prefabricator includes, to the extent pertinent to page prefabrication, a start loader, a benefit analyzer, a URL collector, and a page generator. **Fig. 3** on **p. 3** and **Section 3.3** on **p.12** of **Exhibit B** shows that the Benefit Analyzer receives PRBs to be processed from both the URL Collector and the Start Loader and determines the next set of pages to prefabricate. **Section 3.3** on **p.12** further discloses the selection algorithm of how the Benefit Analyzer determines which set of pages to prefabricate and what information must be made available to the Benefit Analyzer in order for the Benefit Analyzer to serve its intended functions. As **Section 3.3** clearly demonstrates, the selection algorithm of the Benefit Analyzer is based primarily upon the current fabrication rate of the system but need not rely on any user requests to make such determination.

(e) **Section 3.2** describes the Start Loader. Moreover, the logic for the Start Loader to implement the Runnable interface further describes how the Start Loader serves its intended functions. More specifically, the Start Loader uses the PRBEvaluateInterface implementer to generate all the PRBs that make up the initial workload; it then invokes the Benefit Analyzer to populate the PRBs; and the thread periodically wakes up to check if new PRBs need to be

created and creates them. From this description, it is clear that the Start Loader requires no user requests to serve its intended purposes.

(f) **Section 3.4** describes the algorithm for the Page Request Feeder which performs page generation from the PRBs that have been processed by the Benefit Analyzer. As it can be seen from the description, the algorithm requires no user request to serve its intended function.

(g) **Sections 3.5** and **3.6** together with **Figs. 1-3** describe the functions of the URL Collector and the Page Generator. These paragraphs are absolutely silent on taking on any user requests in order to serve the functions. More specifically, **Section 3.5** describes that the URL Collector functions like a web-crawler. It is well know in the art that a web-crawler typically functions in an automated manner. The Page Generator may also function free of any user requests because the Page Generator is interposed between the Page Request Feeder and the URL Collector, both of which may function independently from any user requests.

(2) *wherein the act of prefabricating the first page comprises querying a database to obtain cached data, processing the data received from the database, and packaging information associated with the data in a prescribed format*

(a) Particularly **Section 3.2** explicitly discloses that the functions of the default implementer of the PRBEvaluateInterface including "***query*** the find_user, find_user_resp_groups and find_responsibility tables . . . ." Thus, we believe **Exhibit B** clearly discloses the claimed limitation of "querying a database . . . ."

8. With respect to claim 58:

Exhibit A shows a design specification for prefabricating an information page, that includes

(1) prefabricating a first page to produce a first prefabricated page, wherein the prefabricating is not in response to a request for the first page by a user (particularly **p. 7, Section 3.1.1-3.1.2**, and **p. 13, Section 3.4**),

(2) receiving an information request from a user having a session identifier (particularly **p. 20, Section 3.9**),

(3) determining if the information request corresponds to the first page (particularly **p. 20, Section 3.9**),

(4) providing the first prefabricated page with the session identifier if the information request corresponds to the first page (particularly, **p. 20, Section 3.9**), and

(5) dynamically fabricating a second page if the information request corresponds to the second page (particularly **p. 11, Section 3.2**, and **p. 19, Section 3.9**),

(6) wherein the act of prefabricating the first page comprises querying a database to obtain cached data, processing the data received from the database, and packaging information associated with the data in a prescribed format (particularly **p. 19, Section 3.6**, and **p. 20, Sections 3.8-3.9**).

In addition, **Exhibit B** is part of a presentation made internally within the company of the assignee describing analysis results based upon running an implementation of the invention that includes:

(1) *prefabricating a first page to produce a first prefabricated page, wherein the prefabricating is not in response to a request for the first page by a user*

(a) **p. 4, Section 2** "[t]he key idea behind page prefabrication is to create and store the HTTP Response for a URL request ***a priori***, *so that* ***when an user actually requests the URL it can be serviced immediately***"

(b) **p. 5** of **Exhibit B** shows the three major components of the design to be the prefabricator, the interceptor, and the cache where at least one of the embodiments of the claimed invention as well as **Exhibit B** are focusing upon the interceptor and the prefabricator. **Exhibit B** thus demonstrates that both the interceptor and the prefabricator may serve their respective functions without receiving any user requests.

(c) **Fig. 1** on **p. 4** and **Section 3.9** of **Exhibit B** on **pp. 20-21** show that the interceptor intercepts the URL requests either from the prefabricator or the user and thereby determines whether the request can be serviced from the cache or not. **Section 3.9** further discloses the basic algorithm implemented by the interceptor to include both "[i]f from user" and "[i]f from spider" and "[i]n either case, response should also be T-d to the URL Collector

. . . ." As such, the interceptor can clearly serve its intended function without referencing to any user requests, and **Exhibit B** thus discloses the aforementioned limitations.

(d) **Section 3** of **Exhibit B** shows that the prefabricator includes, to the extent pertinent to page prefabrication, a start loader, a benefit analyzer, a URL collector, and a page generator. **Fig. 3** on **p. 3** and **Section 3.3** on **p.12** of **Exhibit B** shows that the Benefit Analyzer receives PRBs to be processed from both the URL Collector and the Start Loader and determines the next set of pages to prefabricate. **Section 3.3** on **p.12** further discloses the selection algorithm of how the Benefit Analyzer determines which set of pages to prefabricate and what information must be made available to the Benefit Analyzer in order for the Benefit Analyzer to serve its intended functions. As **Section 3.3** clearly demonstrates, the selection algorithm of the Benefit Analyzer is based primarily upon the current fabrication rate of the system but need not rely on any user requests to make such determination.

(e) **Section 3.2** describes the Start Loader. Moreover, the logic for the Start Loader to implement the Runnable interface further describes how the Start Loader serves its intended functions. More specifically, the Start Loader uses the PRBEvaluateInterface implementer to generate all the PRBs that make up the initial workload; it then invokes the Benefit Analyzer to populate the PRBs; and the thread periodically wakes up to check if new PRBs need to be created and creates them. From this description, it is clear that the Start Loader requires no user requests to serve its intended purposes.

(f) **Section 3.4** describes the algorithm for the Page Request Feeder which performs page generation from the PRBs that have been processed by the Benefit Analyzer. As it can be seen from the description, the algorithm requires no user request to serve its intended function.

(g) **Sections 3.5** and **3.6** together with **Figs. 1-3** describe the functions of the URL Collector and the Page Generator. These paragraphs are absolutely silent on taking on any user requests in order to serve the functions. More specifically, **Section 3.5** describes that the URL Collector functions like a web-crawler. It is well know in the art that a web-crawler typically functions in an automated manner. The Page Generator may also function free of

any user requests because the Page Generator is interposed between the Page Request Feeder and the URL Collector, both of which may function independently from any user requests.

(2) *wherein the act of prefabricating the first page comprises querying a database to obtain cached data, processing the data received from the database, and packaging information associated with the data in a prescribed format*

(a) Particularly **Section 3.2** explicitly discloses that the functions of the default implementer of the PRBEvaluateInterface including "***query*** the find_user, find_user_resp_groups and find_responsibility tables . . . ." Thus, we believe **Exhibit B** clearly discloses the claimed limitation of "querying a database . . . ."

9. With respect to claim 63:

Exhibit A shows a design specification for prefabricating an information page, that includes

(1) *obtaining one or more parameters that define how a page should be prefabricated* (particularly **p. 11, Section 3.2,** and **p. 12**), and

(2) *prefabricating a page based on the one or more parameters, wherein the prefabricating is not in response to a request for the page by a user, and wherein the act of prefabricating the page comprises querying a database to obtain cached data, processing the data received from the database, and packaging information associated with the data in a prescribed format* (particularly **p. 13, Section 3.4, p. 19, Section 3.6**, and **p. 21, Section 3.10**).

In addition, **Exhibit B** is part of a presentation made internally within the company of the assignee describing analysis results based upon running an implementation of the invention that includes:

(1) *prefabricating a page based on the one or more parameters, wherein the prefabricating is not in response to a request for the page by a user, and wherein the act of prefabricating the page comprises querying a database to obtain cached data, processing the data received from the database, and packaging information associated with the data in a prescribed format*

(a) **p. 4, Section 2** "[t]he key idea behind page prefabrication is to create and store the HTTP Response for a URL request ***a priori***, *so that* ***when an user actually requests the URL it can be serviced immediately***"

(b) **p. 5 Exhibit B** shows the three major components of the design to be the prefabricator, the interceptor, and the cache where at least one of the embodiments of the claimed invention as well as **Exhibit B** are focusing upon the interceptor and the prefabricator. **Exhibit B** thus demonstrates that both the interceptor and the prefabricator may serve their respective functions without receiving any user requests.

(c) **Fig. 1** on **p. 4** and **Section 3.9** of **Exhibit B** on **pp. 20-21** show that the interceptor intercepts the URL requests either from the prefabricator or the user and thereby determines whether the request can be serviced from the cache or not. **Section 3.9** further discloses the basic algorithm implemented by the interceptor to include both "[i]f from user" and "[i]f from spider" and "[i]n either case, response should also be T-d to the URL Collector . . . ." As such, the interceptor can clearly serve its intended function without referencing to any user requests, and **Exhibit B** thus discloses the aforementioned limitations.

(d) **Section 3** of **Exhibit B** shows that the prefabricator includes, to the extent pertinent to page prefabrication, a start loader, a benefit analyzer, a URL collector, and a page generator. **Fig. 3** on **p. 3** and **Section 3.3** on **p.12** of **Exhibit B** shows that the Benefit Analyzer receives PRBs to be processed from both the URL Collector and the Start Loader and determines the next set of pages to prefabricate. **Section 3.3** on **p.12** further discloses the selection algorithm of how the Benefit Analyzer determines which set of pages to prefabricate and what information must be made available to the Benefit Analyzer in order for the Benefit Analyzer to serve its intended functions. As **Section 3.3** clearly demonstrates, the selection algorithm of the Benefit Analyzer is based primarily upon the current fabrication rate of the system but need not rely on any user requests to make such determination.

(e) **Section 3.2** describes the Start Loader. Moreover, the logic for the Start Loader to implement the Runnable interface further describes how the Start Loader serves its intended functions. More specifically, the Start Loader uses the PRBEvaluateInterface implementer to

generate all the PRBs that make up the initial workload; it then invokes the Benefit Analyzer to populate the PRBs; and the thread periodically wakes up to check if new PRBs need to be created and creates them. From this description, it is clear that the Start Loader requires no user requests to serve its intended purposes.

(f) **Section 3.4** describes the algorithm for the Page Request Feeder which performs page generation from the PRBs that have been processed by the Benefit Analyzer. As it can be seen from the description, the algorithm requires no user request to serve its intended function.

(g) **Sections 3.5** and **3.6** together with **Figs. 1-3** describe the functions of the URL Collector and the Page Generator. These paragraphs are absolutely silent on taking on any user requests in order to serve the functions. More specifically, **Section 3.5** describes that the URL Collector functions like a web-crawler. It is well know in the art that a web-crawler typically functions in an automated manner. The Page Generator may also function free of any user requests because the Page Generator is interposed between the Page Request Feeder and the URL Collector, both of which may function independently from any user requests.

(2) *prefabricating a page based on the one or more parameters, wherein the prefabricating is not in response to a request for the page by a user, and wherein the act of prefabricating the page comprises querying a database to obtain cached data, processing the data received from the database, and packaging information associated with the data in a prescribed format*

(a) Particularly **Section 3.2** explicitly discloses that the functions of the default implementer of the PRBEvaluateInterface including "***query*** the find_user, find_user_resp_groups and find_responsibility tables . . . ." Thus, we believe **Exhibit B** clearly discloses the claimed limitation of "querying a database . . . ."

10. We have reviewed the currently pending claims of the above-identified application (i.e., claims 1-91), and to the best of our recollection, ***we believe that we had a definite and complete idea of, and had actually reduced to practice, the designs embodying all of the elements of claims 1-91 prior to December 18, 2000.*** Dependent claims 2 and 72 are set forth below:

2. (Original) The method of claim 1 further comprising:
determining if the first prefabricated page is stale;
dynamically fabricating the first page if the first prefabricated page is stale.

72. (Previously Presented) The computer program product of claim 70, the process further comprising:
determining if the first prefabricated page is stale;
dynamically fabricating the first page if the first prefabricated page is stale.

Exhibit A shows a design specification for prefabricating an information page, that includes:

(a) determining if the first prefabricated page is stale; [**p. 6, Section 3 Design Description, Fig. 3**, Benefit Analyzer and **p. 12**, Section 3.3 Benefit Analyzer – Selection Algorithm];

(b) dynamically fabricating the first page if the first prefabricated page is stale. [**p. 6**, Section 3 Design Description, Fig. 3, Benefit Analyzer].

In addition, **Exhibit B** is part of a presentation made internally within the company of the assignee describing analysis results based upon running an implementation of the invention that includes:

(a) determining if the first prefabricated page is stale; [**p. 3, JTF Prefab: Fabrication Rate Depth vs. Staleness of Home Page, Fig**];

(b) dynamically fabricating the first page if the first prefabricated page is stale. [**p.1, What is JTF Prefab Solving?** and **p. 3, JTF Prefab Dynamic -> Static Page**].

11. With respect to dependent claims 3 and 27:

Dependent claim 3 is set forth below:

3. (Original) The method of claim 2 in which a time factor is considered in determining whether the first prefabricated page is stale.

Exhibit A shows a design specification for prefabricating an information page, that includes "a time factor is considered in determining whether the first prefabricated page is stale." [**p. 12,**

Section 3.3 Benefit Analyzer – staleness of home pages and **p.13**, ¶ 3 "[i]f home page PRBs are stale beyond an acceptable time . . . ."].

The above evidence also supports claim 27 which recites "*the module prioritizes the list of pages based upon a page prefabrication time parameter.*"

12. With respect to dependent claims 4 and 73:

Dependent claims 4 and 73 are set forth below:

4. (Original) The method of claim 1 further comprising:
crawling the first prefabricated page;
determining if additional pages should be prefabricated; and
prefabricating the additional pages.
73. (Previously Presented) The computer program product of claim 70, the process further comprising:
crawling the first prefabricated page;
determining if additional pages should be prefabricated; and
prefabricating the additional pages.

Exhibit A shows a design specification for prefabricating an information page, that includes:

(a) crawling the first prefabricated page; [**p. 10, URL Collector** and **p. 19, Section 3.5 URL Collector**];
(b) determining if additional pages should be prefabricated; and [**p.10, Benefit Analyzer, p.12, Section 3.3 Benefit Analyzer**
(c) prefabricating the additional pages. [**p. 6, Section 3 Design Description, Fig. 3, Page Generator**, **p. 10, Page Generator**, and **p. 19, Page Generator**].

13. With respect to dependent claim 5:

Dependent claim 5 is set forth below:

5. (Original) The method of claim 4 in which the first page is a start page.

Exhibit A shows a design specification for prefabricating an information page, that includes "*the first page is a start page.*" [**p. 20, Section 3.8 User Session Management, Item 4** – "*the entry*

page (home page)", **p. 12, Section 3.3 Benefit Analyzer**, "*homes pages being always ranked the highest*", and **p. 13, ¶ 1**, "*ensure that home pages will always have the highest weight . . .*"]

14. With respect to dependent claims 6 and 74:

Dependent claims 6 and 74 are set forth below:

6. (Original) The method of claim 1 in which prefabricating the first page comprises:
querying a database for information;
processing the information; and
packaging the processed information into the first prefabricated page.
74. (Previously Presented) The computer program product of claim 70 in which the prefabricating the first page comprises:
querying a database for information;
processing the information; and
packaging the processed information into the first prefabricated page.

Exhibit A shows a design specification for prefabricating an information page, that includes:

(a) *querying a database for information;* [**pp. 11-12, Section 3.2** – "*the default implementer of the PRBEvaluateInterface does the following: query the find_user, find_user_resp_groups and find_responsibility tables . . . .*"]

(b) *processing the information; and* [**p.6, Section 3 Design Description**, Request Feeder, **p. 12, Section 3.3 Benefit Analyzer** – "*process all the user home pages*," and **p. 13, ¶ 2** – "*Periodically . . . issue the next set of PRBs to be processed . . .*"]

(c) *packaging the processed information into the first prefabricated page.* [**p. 7, Section 3.1.1** Page Request Block – homePRB and "*list of constructors*" and "*Who constructs PRBs?*"]

15. With respect to dependent claims 7, 26, and 75:

Dependent claim 7 is set forth below:

7. (Original) The method of claim 1 in which a system resource level is considered before scheduling the action of prefabricating the first page.
26. (Original) The system of claim 25 in which the module prioritizes the list of pages based upon a system resource parameter.

75. (Previously Presented) The computer program product of claim 70 in which the process further comprises considering a system resource level before scheduling the action of prefabricating the first page.

Exhibit A shows a design specification for prefabricating an information page, that includes "*a system resource level is considered before scheduling the action of prefabricating the first page.*" [**pp. 7-8, Section 3.1.2**, Policy Table (PT), and **pp. 8-9, Section 3.1.3**, System Statistics (ST)]

The above evidence also supports claim 26 which recites similar limitations.

16. With respect to dependent claim 8:

Dependent claim 8 is set forth below:

8. (Original) The method of claim 7 in which the system resource level is a resource measure selected from the group consisting of: CPU usage level, memory usage level, and number of pending prefabrication requests.

Exhibit A shows a design specification for prefabricating an information page, that includes "*the system resource level is a resource measure selected from the group consisting of: CPU usage level, memory usage level, and number of pending prefabrication requests.*" [**pp. 7-8, Section 3.1.2**, Policy Table (PT), and **pp. 8-9, Section 3.1.3**, System Statistics (ST)]

17. With respect to dependent claims 9, 64, and 76:

Dependent claims 9 and 64 are set forth below:

9. (Original) The method of claim 1 in which the definable prefabrication policy applies to a specific user or class of users.
64. (Previously Presented) The method of claim 63, wherein one of the one or more parameters is based on a specific user or class of users.
76. (Previously Presented) The computer program product of claim 70 in which the definable prefabrication policy applies to a specific user or class of users.

Exhibit A shows a design specification for prefabricating an information page, that includes "definable prefabrication policy applies to a specific user or class of users." [**pp. 7-8, Sections 3.1.1**

- **3.1.2**, Page Request Block (PRB) and Policy Table (PT) and **pp. 11-12**, Start Loader – default implementer of the PRBEvaluateInterface class]

18. With respect to dependent claims 10 and 77:

Dependent claims 10 and 77 are set forth below:

> *10. (Original) The method of claim 1 in which the definable prefabrication policy identifies pages to prefabricate.*
> *77. (Previously Presented) The computer program product of claim 70 in which the definable prefabrication policy identifies pages to prefabricate.*

Exhibit A shows a design specification for prefabricating an information page, that includes "the definable prefabrication policy identifies pages to prefabricate." [**pp. 7-8, Section 3.1.2**, Section 3.1.2 Policy Table (PT) – Depth].

19. With respect to dependent claim 11:

Dependent claim 11 is set forth below:

> *11. (Original) The method of claim 10 in which the definable prefabrication policy comprises a responsibility parameter.*

Exhibit A shows a design specification for prefabricating an information page, that includes "*the definable prefabrication policy comprises a responsibility parameter*". [**pp. 7-8, Section 3.1.2**, Section 3.1.2 Policy Table (PT) – Responsibilities].

20. With respect to dependent claims 12 and 66:

Dependent claims 12 and 66 are set forth below:

> *12. (Original) The method of claim 10 in which the definable prefabrication policy comprises an application identifier.*
> *66. (Previously Presented) The method of claim 63, further comprising identifying an application for which the page should be prefabricated based on the one or more parameters.*

Exhibit A shows a design specification for prefabricating an information page, that includes "*the definable prefabrication policy comprises an application identifier.*" [**p. 7**, Section 3.1.1 Page Request Block (PRB) – "appID".]

21. With respect to dependent claims 13 and 67:

Dependent claims 13 and 67 are set forth below:

> *13. (Original) The method of claim 10 in which the definable prefabrication policy comprises a scheduling parameter.*
> *67. (Previously Presented) The method of claim 63, wherein one of the one or more parameters comprises a scheduling parameter.*

Exhibit A shows a design specification for prefabricating an information page, that includes "*the definable prefabrication policy comprises a scheduling parameter.*" [**pp. 7-8, Section 3.1.2,** Policy Table (PT)].

22. With respect to dependent claims 14 and 68:

Dependent claims 14 and 68 are set forth below:

> *14. (Original) The method of claim 10 in which the definable prefabrication policy comprises a refresh rate parameter.*
> *68. (Previously Presented) The method of claim 63, wherein one of the one or more parameters comprises a refresh rate parameter.*

Exhibit A shows a design specification for prefabricating an information page, that includes "*the definable prefabrication policy comprises a refresh rate parameter.*" [**pp. 7-8, Section 3.1.2,** Policy Table (PT) – Refresh Interval].

23. With respect to dependent claims 15 and 78:

Dependent claims 15 and 78 are set forth below:

> *15. (Original) The method of claim 1 in which auto-tuning of the prefabricating step is performed to minimize interference with other system workload.*

78. (Previously Presented) The computer program product of claim 70 in which the step of prefabricating comprises performing auto-tuning to minimize interference with other system workload.

Exhibit A shows a design specification for prefabricating an information page, that includes "*auto-tuning of the prefabricating step is performed to minimize interference with other system workload.*" [**pp. 7-8**, Section 3.1.2, Policy Table (PT) and **pp. 8-9**, Section 3.1.3 System Statistics (ST)]

24. With respect to dependent claims 16, 69, and 79:

Dependent claims 16, 69, and 79 are set forth below:

16. (Original) The method of claim 1 in which the definable prefabrication policy is organized as a hierarchy of policies.
69. (Previously Presented) The method of claim 63 , wherein the one or more parameters are organized as a hierarchy of policy categories.
79. (Previously Presented) The computer program product of claim 70 in which the definable prefabrication policy is organized as a hierarchy of policies.

Exhibit A shows a design specification for prefabricating an information page, that includes "*the definable prefabrication policy is organized as a hierarchy of policies.*" [**p. 6**, Section 3, Design Description – PolicyInformation class, **pp. 7-8**, Section 3.1.2, Policy Table (PT), and **p. 23**, Section 4 Internal APIs/Classes]

25. With respect to dependent claim 17:

Dependent claim 17 is set forth below:

17. (Original) The method of claim 16 in which the definable prefabrication policy comprises a system policy.

Exhibit A shows a design specification for prefabricating an information page, that includes "*the definable prefabrication policy comprises a system policy.*" [**pp. 7-9**, Sections 3.1.2 – 3.1.3]

26. With respect to dependent claim 18:

Dependent claim 18 is set forth below:

> *18. (Original) The method of claim 16 in which the definable prefabrication policy comprises an application policy.*

Exhibit A shows a design specification for prefabricating an information page, that includes "*the definable prefabrication policy comprises an application policy.*" [pp. 7-8, Section 3.1.2, Policy Table – table including "AppID".]

27. With respect to dependent claim 19:

Dependent claim 19 is set forth below:

> *19. (Original) The method of claim 16 in which the definable prefabrication policy comprises a user policy.*

Exhibit A shows a design specification for prefabricating an information page, that includes "*the definable prefabrication policy comprises a user policy.*" [**pp. 9-11**, Section 3.1.4 User Page Table; **p. 7**, Section 3.1.1 Page Request Block (PRB); and **pp. 7-8**, Section 3.1.2, Policy Table (PT).]

28. With respect to dependent claim 20:

Dependent claim 20 has been cancelled.

29. With respect to dependent claims 21, 36, and 80:

Dependent claims 21, 36, and 80 are set forth below:

> *21. (Original) The method of claim 1 in which the first page comprises a browser page.*
> *36. (Original) The system of claim 23 in which the computer application comprises a database application.*
> *80. (Previously Presented) The computer program product of claim 70 in which the first page comprises a browser page.*

Exhibit A shows a design specification for prefabricating an information page, that includes "*the first page comprises a browser page.*" [**p. 4**, Section 2, High Level Design].

The above evidence also supports claim 36 which recites similar limitations.

30. With respect to dependent claims 22 and 81:

Dependent claims 22 and 81 is set forth below:

> *22. (Original) The method of claim 1 in which the first prefabricated page is cached.*
> *81. (Previously Presented) The computer program product of claim 70 in which the process further comprises caching the first prefabricated page.*

Exhibit A shows a design specification for prefabricating an information page, that includes "*the first prefabricated page is cached.*" [**p. 4**, Section 2, High Level Design, **p. 19**, Section 3.5 URL Collector, and **p. 5**, Design].

31. With respect to dependent claim 24:

Dependent claim 24 is set forth below:

> *24. (Original) The system of claim 23 in which the prefabricator comprises a module to identify pages to prefabricate.*

Exhibit A shows a design specification for prefabricating an information page, that includes "*the prefabricator comprises a module to identify pages to prefabricate.*" [**p. 10**, Benefit Analyzer, and **pp. 12-13**, Benefit Analyzer].

32. With respect to dependent claim 25:

Dependent claim 25 is set forth below:

> *25. (Original) The system of claim 23 in which the prefabricator comprises a module to prioritize a list of pages to prefabricate.*

Exhibit A shows a design specification for prefabricating an information page, that includes "*the prefabricator comprises a module to prioritize a list of pages to prefabricate..*" [**p. 10**, Benefit Analyzer, **pp. 12-13**, Benefit Analyzer, and **p. 14**, Open].

33. With respect to dependent claim 26:

Dependent claim 25 is set forth below:

> *25. (Original) The system of claim 23 in which the prefabricator comprises a module to prioritize a list of pages to prefabricate.*

Exhibit A shows a design specification for prefabricating an information page, that includes "*the prefabricator comprises a module to prioritize a list of pages to prefabricate.*" [**p. 10**, Benefit Analyzer, **pp. 12-13**, Benefit Analyzer, and **p. 14**, Open].

34. With respect to dependent claim 28:

Dependent claim 26 is set forth below:

> *28. (Original) The system of claim 25 in which the module prioritizes the list of pages based upon a user access pattern parameter.*

Exhibit B is part of a presentation made internally within the company of the assignee describing analysis results based upon running an implementation of the invention that includes "*the module prioritizes the list of pages based upon a user access pattern parameter.*" [**p. 5**, JTF Prefab Service Algorithm].

35. With respect to dependent claim 29:

Dependent claim 29 is set forth below:

> *29. (Original) The system of claim 25 in which the module prioritizes the list of pages based upon a page depth parameter.*

Exhibit A shows a design specification for prefabricating an information page, that includes "*the module prioritizes the list of pages based upon a page depth parameter.*" [**p. 6**, Section 3, Design Description, and **pp. 7-9**, Section 3.1 Prefabricator Data Structures].

36. With respect to dependent claim 30:

Dependent claim 30 is set forth below:

> *30. (Original) The system of claim 23 in which the first page corresponds to a page request, wherein the page request is processed as a second information request to the interceptor.*

Exhibit A shows a design specification for prefabricating an information page, that includes "*the first page corresponds to a page request, wherein the page request is processed as a second information request to the interceptor.*" [**p. 6**, Section 3, Design Description].

37. With respect to dependent claim 31:

Dependent claim 31 is set forth below:

> *31. (Original) The system of claim 30 in which the prefabricator comprises a module to determine a number of page requests to concurrently process into prefabricated pages.*

Exhibit A shows a design specification for prefabricating an information page, that includes "*the prefabricator comprises a module to determine a number of page requests to concurrently process into prefabricated pages.*" [**p. 7**, Section 3.1.1 Page Request Block (PRB), wherein the "depth" attribute describes the process of "starting from the home page, the number of pages to be navigated before this URL is reached; **pp. 7-11**, Section 3.1 Prefabricator Data Structures.].

38. With respect to dependent claim 33:

Dependent claim 33 is set forth below:

> *33. (Original) The system of claim 23 in which the prefabricator comprises a module to crawl the first prefabricated page for additional pages to prefabricate.*

Exhibit A shows a design specification for prefabricating an information page, that includes "*the prefabricator comprises a module to crawl the first prefabricated page for additional pages to prefabricate.*" [**p. 10** and **p. 19**, URL Collector].

39. With respect to dependent claim 34:

Dependent claim 34 is set forth below:

> *34. (Original) The system of claim 23 in which the prefabricator accesses a prefabrication policy to manage prefabricating the first page.*

Exhibit A shows a design specification for prefabricating an information page, that includes "*the prefabricator accesses a prefabrication policy to manage prefabricating the first page.*" [**pp. 6-8**, Sections 3 – 3.1.2].

40. With respect to dependent claim 36:

Dependent claim 36 is set forth below:

> *36. (Original) The system of claim 23 in which the computer application comprises a database application.*

Exhibit A shows a design specification for prefabricating an information page, that includes "*the computer application comprises a database application.*" [**pp. 7-8**, Sections 3.1.2].

41. With respect to dependent claim 37:

Dependent claim 37 is set forth below:

> *37. (Original) The system of claim 23 in which the interceptor is integrated into a web server.*

Exhibit A shows a design specification for prefabricating an information page, that includes "*the interceptor is integrated into a web server.*" [**p. 3**, Section 3 Design Description and **p. 20**

Section 3.9 Interceptor – where either the Interceptor or the web server receives the requests to get the response generated and thus the Interceptor component may inherently be integrated into the web server].

42. With respect to dependent claim 38:

Dependent claim 38 is set forth below:

> *38. (Original) The system of claim 23 in which the interceptor is integrated with a cache server.*

Exhibit A shows a design specification for prefabricating an information page, that includes "*the interceptor is integrated with a cache server.*" [**p. 20** Section 3.9 Interceptor – where the Interceptor receives the requests and checks if the requests exist in the cache and where the requests originate from the spider or are not present in cache, service the page(s) and store responses in the cache. Thus, the Interceptor component is integrated with a cache server].

43. With respect to dependent claim 39:

Dependent claim 39 is set forth below:

> *39. (Original) The system of claim 23 in which the prefabricator comprises a module to monitor system resources.*

Exhibit A shows a design specification for prefabricating an information page, that includes "*the prefabricator comprises a module to monitor system resources.*" [**pp. 7-8, Section 3.1.2**, Policy Table (PT), and **pp. 8-9, Section 3.1.3**, System Statistics (ST). Where the system resource level is considered before scheduling the prefabrication and thus there must exist a system resource monitoring module so as to obtain and thereby consider the system resource level.]

44. With respect to dependent claim 40:

Dependent claim 40 is set forth below:

40. (Original) The system of claim 23 in which the prefabricator and the interceptor are logically associated with a first network node, wherein the system further comprises:

a second prefabricator and a second interceptor logically associated with a second network node.

Exhibit A shows a design specification for prefabricating an information page, that includes "*a second prefabricator and a second interceptor logically associated with a second network node.*" [**pp. 4-5**, Section 2 High Level Design].

45. With respect to dependent claim 41-46, 50, 54-57, 83, and 86:

Dependent claims 41-46, 50, 54-57, 83, and 86 are set forth below:

41. (Original) The system of claim 40 in which the routing component routes information requests among the first and second network nodes.
42. (Original) The system of claim 40 in which a load distributor distributes a prefabrication workload among the first and second network nodes.
43. (Original) The system of claim 42 in which the prefabrication workload is distributed based upon system resource levels at the first and second network nodes.
44. (Previously Presented) The system of claim 43 in which a node is assigned a share of the prefabrication workload based on a resource level of the node.
45. (Original) The system of claim 43 in which each of the first and second network nodes are assigned work from the prefabrication workload based upon its individual resource levels without regard to resource levels on other nodes.
46. (Original) The system of claim 43 in which the first and second network nodes are assigned work from the prefabricated workload in a coordinated manner.
50. (Original) The method of claim 49 further comprising:
routing the information request to either the first or second node.
54. (Original) The method of claim 49 in which a prefabrication workload is distributed among the first and second nodes.
55. (Previously Presented) The method of claim 54 in which a node is assigned a share of the prefabrication workload based on a resource level of the node.
56. (Original) The method of claim 54 in which each of the first and second nodes are assigned work from the prefabrication workload based upon its individual resource levels without regard to resource levels on other nodes.
57. (Original) The method of claim 54 in which the first and second nodes are assigned work from the prefabricated workload in a coordinated manner.
83. (Previously Presented) The computer program product of claim 71, the process further comprising:
routing the information request to either the first or second node.

86. (Previously Presented) The computer program product of claim 71 in which the process further comprises distributing a prefabrication workload among the first and second nodes.

It is common knowledge for one skilled in the art at the time of invention to distribute workload and thus route requests among multiple processing nodes. It is also common knowledge for one skilled in the art that the distribution of workload is based on system resource level and workload. It is also common knowledge for one skilled in the art that the distribution of tasks may be based on the system resource level of the node to which the tasks are assigned but not the resource levels of other nodes which are irrelevant to the determination of whether to distribute tasks to a particular node. Moreover, it is also common knowledge to distribute workload in a coordinated manner.

46. With respect to dependent claims 47, 52, and 85:

Dependent claims 47, 52, and 85 are set forth below:

47. (Original) The system of claim 40 in which prefabricated pages are stored in a network accessible storage device.
52. (Original) The method of claim 49 in which the first and second prefabricated pages are stored on a network accessible storage device.
85. (Previously Presented) The computer program product of claim 71 in which the process further comprises storing the first and second prefabricated pages on a network accessible storage device.

Exhibit A shows a design specification for prefabricating an information page, that includes "*prefabricated pages are stored in a network accessible storage device.*" [**pp. 4-5**, Section 2 High Level Design and **p. 19**, Section 3.5 URL Collector, **pp. 20-21**, Section 3.9 Interceptor, and **pp. 21-22**, Section 3.10 Cache].

47. With respect to dependent claim 48:

Dependent claim 48 is set forth below:

48. (Original) The system of claim 23 which is non-intrusively implemented with an existing computer application such that code changes are not performed against the existing computer application.

Exhibit B Exhibit B is part of a presentation made internally within the company of the assignee describing analysis results based upon running an implementation of the invention that includes "*non-intrusively implemented with an existing computer application such that code changes are not performed against the existing computer application.*" [**p. 2**, JTF Prefab Applications (OSO) Should Provide].

48. With respect to dependent claims 51 and 84:

Dependent claims 51 and 84 are set forth below:

51. (Original) The method of claim 49 in which the first node accesses the second prefabricated page to satisfy the information request.
84. (Previously Presented) The computer program product of claim 71 in which the process further comprises causing the first node to access the second prefabricated page to satisfy the information request.

Exhibit A shows a design specification for prefabricating an information page, that includes "*the first node accesses the second prefabricated page to satisfy the information request.*" [**pp. 7-8**, Sections 3.1.1 – 3.1.2 "depth"].

Exhibit B Exhibit B is part of a presentation made internally within the company of the assignee describing analysis results based upon running an implementation of the invention that includes "*the first node accesses the second prefabricated page to satisfy the information request.*" [**p. 3**, JTF Prefab: Fabrication Rate Depth vs. Staleness of Home Page]

49. With respect to dependent claim 53:

Dependent claim 53 is set forth below:

53. (Original) The method of claim 52 in which network accessible storage device comprises a NFS-compliant device.

Applicants respectfully submit that NFS (Network File System) was developed in mid-1980s and constitutes common knowledge which one of ordinary skill in the art certainly possesses in devising and using a network accessible storage device.

50. With respect to dependent claim 59:

Dependent claim 59 is set forth below:

59. (Original) The method of claim 58 further comprising:
verifying validity of the session identifier.

Exhibit A shows a design specification for prefabricating an information page, that includes "*verifying validity of the session identifier.*" [**p. 6**, Invalidator, **pp. 19-20**, Section 3.7 Page Invalidator, and **p. 20**, Section 3.8 User Session Management].

Exhibit B Exhibit B is part of a presentation made internally within the company of the assignee describing analysis results based upon running an implementation of the invention that includes "*verifying validity of the session identifier.*" [**p. 6**, JTF Prefab Invalidation Policy]

51. With respect to dependent claim 60-61:

Dependent claims 60 and 61 are set forth below:

60. (Original) The method of claim 59 further comprising:
distributing a message verifying the validity of the session identifier to one or more network nodes.
61. (Original) The method of claim 58 in which the session identifier is provided with the first prefabricated page as a URL parameter.

Exhibit A shows a design specification for prefabricating an information page, that includes "*distributing a message verifying the validity of the session identifier to one or more network nodes.*" [**pp. 19-20**, Section 3.7 Page Invalidator, and **p. 20**, Section 3.8 User Session Management].

52. With respect to dependent claim 62:

Dependent claim 62 is set forth below:

62. (Original) The method of claim 58 in which the session identifier is provided with the first prefabricated page as a cookie value.

Exhibit A shows a design specification for prefabricating an information page, that includes "*the session identifier is provided with the first prefabricated page as a cookie value.*" **[pp. 19-21,** Sections 3.7-3.9].

53. With respect to dependent claim 65:

Dependent claim 65 is set forth below:

65. (Previously Presented) The method of claim 63, further comprising identifying a page to prefabricate based on the one or more parameters.

Exhibit A shows a design specification for prefabricating an information page, that includes "*identifying a page to prefabricate based on the one or more parameters.*" **[pp. 7-8, Section 3.1.2** shows several parameters for pages to prefabricate.]

54. With respect to dependent claims 82 and 87-91:

Dependent claims 82 and 87-88 are set forth below:

82. (Previously Presented) The computer program product of claim 70, wherein the prefabricating is performed in response to a request initiated by a software, a hardware, or a combination of both.
87. (Previously Presented) The computer program product of claim 71, wherein the prefabricating is performed in response to a request initiated by a software, a hardware, or a combination of both.
88. (Previously Presented) The method of claim 1, wherein the prefabricating is performed in response to a request initiated by a software, a hardware, or a combination of both.
89. (Previously Presented) The method of claim 49, wherein the prefabricating is performed in response to a request initiated by a software, a hardware, or a combination of both.
90. (Previously Presented) The method of claim 58, wherein the prefabricating is performed in response to a request initiated by a software, a hardware, or a combination of both.

91. (Previously Presented) The method of claim 63, wherein the prefabricating is performed in response to a request initiated by a software, a hardware, or a combination of both.

Exhibit A shows a design specification for prefabricating an information page, that includes "*prefabricating is performed in response to a request initiated by a software, a hardware, or a combination of both.*" [**p. 4**, Section 2, High Level Design and **pp. 11-12**, Section 3.2 Start Loader – Implementation Details]

55. The subject invention was reduced to practice and tested to verify that it works for its intended purpose prior to December 18, 2000. **Exhibit B** includes information which evidences that the subject invention was tested and found to work for its intended purpose.

More specifically, the last page of **Exhibit A** titled "*Response Time For Home Page*" demonstrated one example that the response time for a given home page was improved from 3.76 seconds, done without prefabrication, to 0.53 seconds with prefabrication. **Exhibits A** also presents various graphs demonstrating the variations of the PRF (Page Request Feeder) Q time and process time as well as the Refresh Time corresponding to different thread counts. All these graphs and results as shown in **Exhibit A** clearly demonstrate that **not only did the claimed invention exist prior to December 18, 2000, but it was actually implemented and tested and therefore reduced to practice, also prior to December 18, 2000**, in various ways to verify that the claimed invention serves its intended purpose of improving page retrieval time by using the claimed invention. Furthermore, our recognition of one of the goals of the claimed invention, i.e., to improve page retrieval time by using page prefabrication, together with the **actual improvement in response time from 3.76 seconds to 0.53 seconds** actually demonstrated that **we appreciated and recognized the claimed invention prior to December 18, 2000**.

56. We declare that all statements made herein of our own knowledge are true, and that all statements made on information and belief are believed *to* be true; and further, that these statements were made with the knowledge that willful false statements and the like so made are punishable by fine or imprisonment, or both, Section 1001. of Title 18 of United States Code, and that such willful false statements may jeopardize the validity of the application or any patent issuing thereon.

Date	Sowmya Subramanian
Date	Ramu Sunkara
Date	Kunal Kapur
Date	Anthony Lai
Date	Sarim Siddiqui
Date	Sunny Wong
Date	Hyun-Sik Byun

May 16, 2007
Date

Sowmya Subramanian
Sowmya Subramanian

Date

Ramu Sunkara

Date

Kunal Kapur

Date

Anthony Lai

Date

Sarim Siddiqui

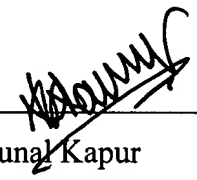
Date

Sunny Wong

Date

Hyun-Sik Byun

Date	Sowmya Subramanian
MAY 15 2007 Date	Ramu V. Sunkara Ramu Sunkara
Date	Kunal Kapur
Date	Anthony Lai
Date	Sarim Siddiqui
Date	Sunny Wong
Date	Hyun-Sik Byun

Date	Sowmya Subramanian
Date	Ramu Sunkara
5/17/2007 Date	 Kunal Kapur
Date	Anthony Lai
Date	Sarim Siddiqui
Date	Sunny Wong
Date	Hyun-Sik Byun

Date	Sowmya Subramanian
Date	Ramu Sunkara
Date	Kunal Kapur
4/11/07 Date	Anthony Lai Anthony Lai
Date	Sarim Siddiqui
Date	Sunny Wong
Date	Hyun Byun

Date

Sowmya Subramanian

Date

Ramu Sunkara

Date

Kunal Kapur

Date

Anthony Lai

04/09/07

Date

Sarim Siddiqui

Date

Sunny Wong

Date

Hyun-Sik Byun

Patent
262/118
OI7011452001

Date	Sowmya Subramanian
Date	Ramu Sunkara
Date	Kunal Kapur
Date	Anthony Lai
Date	Sarim Siddiqui
Date	Sunny Wong
Date	Hyun-Sik Byun

Date	Sowmya Subramanian
Date	Ramu Sunkara
Date	Kunal Kapur
Date	Anthony Lai
Date	Sarim Siddiqui
Date	Sunny Wong
5/17/07 Date	Hyun-Sik Byun